TECNICHE DI PROGRAMMAZIONE

11/06/19

Tema d'esame "Serie A" seconda parte.

In questo esercizio avevamo già completato i punti “a” e “b” del primo punto. Passiamo quindi allo sviluppo del punto “c”.

Immagine che contiene screenshot

Descrizione generata automaticamente

Al momento, nel Model, abbiamo le informazioni sulle squadre, sulle stagioni e sui punteggi a fine stagione. Con queste informazioni, ci dobbiamo costruire un grafo in cui i vertici rappresentino le stagioni e gli archi ci dicano se la determinata squadra selezionata abbia fatto più punti in una stagione piuttosto che in un’altra.

Gli archi devono quindi essere orientati e pesati. Creiamo quindi il grafo nel Model e lo andiamo a costruire quando dall’esterno ci viene chiesto di trovare l’annata d’oro.

Andiamo quindi prima a creare il metodo “calcolaAnnataDOro” (sempre nel Model) che ci restituirà una Season. In questo metodo noi andremo a costruire il grafo e a trovare l’annata migliore. Per quanto riguarda la squadra selezionata, l’abbiamo salvata nella variabile “squadraSelezionata” nel metodo “calcolaPunteggi”.

Immagine che contiene screenshot

Descrizione generata automaticamente

Nella costruzione del grafo, creiamo un nuovo oggetto di tipo grafo (“SimpleDirectedWeightedGraph ”). Una volta creato, andiamo ad aggiungere i vertici ad esso mediante il metodo “addAllVertices” della classe “Graphs”. Per trovare gli archi devo andare a cercare tutte le combinazioni di stagioni (utilizzo 2 “for”). Controllo che le 2 stagioni siano diverse. I punteggi per ogni stagione li ho nella Mappa “punteggi”. Me li salvo in 2 variabili diverse per le due stagioni e poi controllo che i punteggi di una stagione siano maggiori di quelli dell’altra. In caso positivo, creo l’arco corrispondente con il metodo “addEdge”. In caso negativo, creo l’arco nell’altra direzione perché la squadra ha fatto più punti nell’altra stagione.

Immagine che contiene screenshot

Descrizione generata automaticamente

La parte sulla creazione del grafo è terminata; passiamo ora alla ricerca dell’annata migliore cioè della stagione in cui la somma dei pesi degli archi entranti meno la somma degli archi uscenti sia massima.

Prima però abbiamo trovato un errore; nel grafo noi aggiungiamo tutte le stagioni quando in realtà dobbiamo inserire solo le stagioni in cui la squadra selezionata ha giocato. Nel metodo “addAllVertices”, al posto di usare la lista “stagioni” che contiene tutte le stagioni, usiamo le chiavi della mappa punteggi (un sottoinsieme di stagioni in cui la squadra ha giocato).

Per trovare l’annata migliore incominciamo con l’inizializzare a “null” un oggetto “Season” chiamata “migliore” che rappresenta appunto la stagione migliore e a “0” la variabile “max” che andrà a contenere il punteggio massimo. Successivamente itero su tutti i vertici del grafo e per ogni vertice mi calcolo il peso.

Per far questo uso un metodo nuovo chiamato “pesoStagione” (andiamo a crearlo dopo). Nel caso in cui il peso trovato da questo metodo sia maggiore di “max”, aggiorno il valore e aggiorno la stagione migliore. Alla fine di questo ciclo potrò restituire la stagione migliore.

Immagine che contiene testo

Descrizione generata automaticamente

Ora andiamo a creare il metodo privato “pesoStagione” a cui passeremo la stagione. In esso dovremo fare la somma dei pesi degli archi entranti meno la somma dei pesi degli archi uscenti. Imposto una variabile denominata “somma” uguale a zero. Grazie al metodo “incomingEdgesOf” (che mi restituisce un Set di archi) trovo tutti gli archi entranti. Per gli archi uscenti posso usare il metodo “outgoingEdgesOf”. Una volta trovato questi due insiemi, posso iterare sia tra gli archi entranti e sommare il loro peso e iterare tra quelli uscenti e sottrare il loro peso alla somma.

Immagine che contiene screenshot

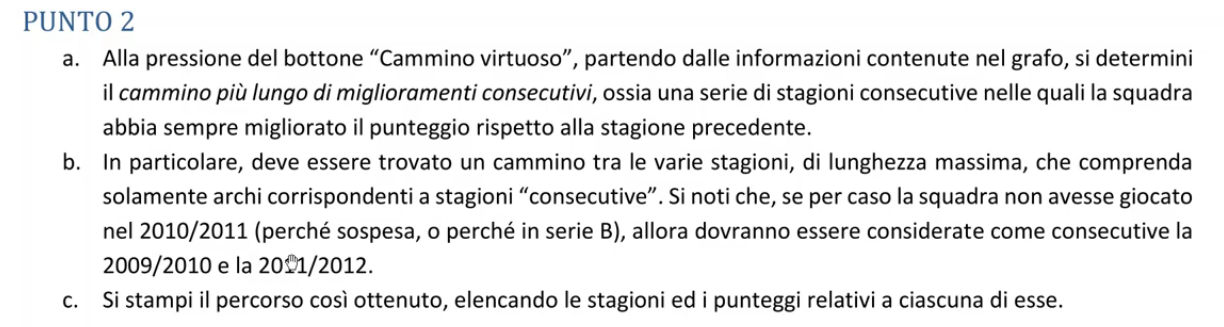
Descrizione generata automaticamente

Ora possiamo collegare il metodo “calcolaAnnataDOro” al Controller. Vado quindi nel metodo “doTrovaAnnataOro” e creo un oggetto “Season” chiamato “annata” in cui inserirò il risultato trovato dal metodo del Model. Come ultima cosa, stampo il risultato ottenuto.

Immagine che contiene screenshot

Descrizione generata automaticamente

Passiamo ora al punto “2”.



Per risolvere questo punto ci serve come prima cosa, la costruzione della sequenza ordinata di annate. Dovremo utilizzare la ricorsione!

Andiamo nel Model e creiamo un metodo denominato “camminoVirtuoso” che ha come obiettivo quello di:

1. Trovare le stagioni consecutive: Creo una lista di “Season” nel Model e la popolo con le stagioni che ho messo nella mappa. Successivamente ordino la lista on il metodo “sort” di “Collections”. Dobbiamo ricordarci di andare nella classe Season e di mettere il Comparable.
2. Preparare le variabili utili alla ricorsione: inizializziamo la lista parziale e il “percorsoBest”.
3. Iterare al livello zero della ricorsione: inserisco i vari vertici e per ognuno di essi provo ad avviare la ricorsione (ricordarsi di rimuovere poi il vertice).

Questo metodo dovrà poi restituire il percorso migliore. La sua struttura sarà la seguente:

Immagine che contiene screenshot

Descrizione generata automaticamente

Per quanto riguarda la ricorsione, la soluzione parziale sarà una lista di stagioni (lista di vertici del grafo). Il livello della ricorsione sarà la lunghezza della lista. Il caso terminale si ha quando non trovo altri vertici da aggiungere. Se succede questo, bisogna verificare se il cammino ha lunghezza massima.

Per la generazione delle diverse soluzioni parziali, dovrò inserire i vertici connessi all’ultimo vertice del percorso (arco orientato nel verso giusto), non ancora considerati e che sono consecutivi.

Andiamo quindi a scrivere la procedura ricorsiva vera e propria che riceverà in ingresso il livello e una lista di “parziale”. In questo metodo, il caso terminale non posso gestirlo all’inizio perché non so se troverò altri vertici da aggiungere. Quindi all’inizio genero nuove soluzioni e poi alla fine valuto il caso terminale.

Iniziamo prendendo l’ultimo vertice della lista e trovando i vertici adiacenti ad esso (usiamo la classe Graphs che ha il metodo “successorListOf”). Una volta fatto questo, itero tra tutti i successori e provo a cercare il prossimo vertice (devo essere sicuro che la soluzione parziale non contenga già il “prossimo”). Devo controllare che “ultimo” e “prossimo” siano consecutivi cioè che all’interno della lista “stagioniConsecutive” abbiano indici consecutivi. Se questi controlli hanno esito positivo, ho trovato il candidato alla lista parziale, lo aggiungo, richiamo la procedura ricorsiva con il livello+1 e lo tolgo.

Il metodo finale della ricorsione è il seguente:

Immagine che contiene screenshot

Descrizione generata automaticamente

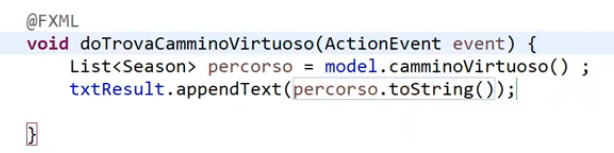
In questo metodo aggiungiamo un FLAG (variabile booleana impostata a false), che diventa true nel caso in cui viene trovato un candidato.

Il caso terminale avviene se questa variabile rimane uguale a “false”. Nel caso in cui ciò accadesse, se la soluzione parziale è più lunga della soluzione migliora trovata fino ad ora, allora ho trovato il nuovo “best”. Ricordarsi sempre che bisogna CLONARE la struttura dati che ha vinto il confronto.

Immagine che contiene screenshot

Descrizione generata automaticamente

Colleghiamo questo metodo al Controller mediante il metodo “doTrovaCamminoVirtuoso”.



FINE.